1. Driver functions
   1. gdennany\_driver: Driver function that calls all the helper functions I have created. Takes as input the raw image and outputs the cleaned image. This function shows the control flow of the entire algorithm.
      1. NOTE: on step 8, you might have to change multiplying the multiplying by 40 or 50 steps to a maximum output value from one of the noise removal functions, but these hardcoded values worked for all the images I have tested it on
   2. gdennany\_driver\_automated: this is where you can automate running the code. The way I have it set up now will let you run the code on a folder of data at a time and output a text file to the current file. You will need to figure out how to run the code and append the outputs to the existing netcdf files from the massachusetts university.
2. Noise Removal and normalization functions
   1. gdennany\_first\_all\_negatives: this function handles the removal of noise and normalization (code has to be normalized to a 0 to 1 scale for Professor Chans inpainting algorithm to work) of clear sky or light precipitation levels. All the data points are negative in these situations.
   2. gdennany\_first\_with\_positives: this function also handles the removal of noise and normalization, but in cases of high precipitation. These cases data points have positive values, and these positive values made the other noise removal technique not work, so I had to make a separate function to handle these types of cases.
3. Filling in holes of removed noise
   1. gdennany\_inpaint: this function basically just takes the output from one of the noise removal functions and runs it through professor Chans inpainting method. In some cases, this method is sufficient and fills in all the removed noise holes. In other cases, the holes are too large and can’t completely fill them in.
   2. gdennany\_get\_new\_binary\_mask: This function takes in whatever input image you give it and outputs a new binary mask for the input image. I used it in the driver function after using Chan’s inpainting method once to get a new mask to use for inward interpolation of the image.
   3. gdennany\_inward\_interpolation: this function takes the inpainted image and a new binary mask for the inpainted image and uses inward interpolation to fill in any remaining holes in the data. The function resizes the image to half the size, does inward interpolation on the half-sized image, and then reverts the image back to the original size. The image is then used in the driver function to “copy” the inwardly interpolated data onto the remaining holes of the inpainted image.
4. Notes:
   1. This algorithm has worked for all the test cases I have tried, but there are a lot of data sets, and I am not sure if it will work for all of them. If there are issues, the problem is most likely in the “noise removal and normalization functions” hard coded maximum and minimum values. To fix this, I would try finding the maximum and minimum of the specific data set and using those values instead of just using +-40 or 50. Upon changing this, don’t forget to change reverting back to the original magnitude in step 8 of the driver function (I mentioned this above as well under the driver description).